ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-1467

Declaratively Capturing Local Label Correlations with Multi-Label Trees

Reem Al-Otaibi [1 2] and **Meelis Kull** [1 3] and **Peter Flach** [1]

Abstract. The goal of multi-label classification is to predict multiple labels per data point simultaneously. Real-world applications tend to have high-dimensional label spaces, employing hundreds or even thousands of labels. While these labels could be predicted separately, by capturing label correlation we might achieve better predictive performance. In contrast with previous attempts in the literature that have modelled label correlations globally, this paper proposes a novel algorithm to model correlations and cluster labels locally. **LaCovaC** is a multi-label decision tree classifier that clusters labels into several dependent subsets at various points during training. The clusters are obtained locally by identifying the conditionally-dependent labels in localised regions of the feature space using the label correlation matrix. **LaCovaC** interleaves between two main decisions on the label matrix with training instances in rows and labels in columns: splitting this matrix vertically by partitioning the labels into subsets, or splitting it horizontally using features in the conventional way. Experiments on 13 benchmark datasets demonstrate that our proposal achieves competitive performance over a wide range of evaluation metrics when compared with the state-of-the-art multi-label classifiers.

1 Introduction and Motivation

In traditional classification each data point is assigned to a single label. In binary classification each point can belong to one of two classes, whereas in multi-class classification the setting is more general, allowing each training point to belong to one of more than two classes. Multi-label classification generalises both by allowing more than one label for each data point [20, 28]. Thus, it allows for a wide range of applications, such as text categorisation, image and movie tagging, and gene function prediction. For example, a medical diagnosis might find a patient has multiple diseases at one time; an article that gives statistics about the number of students who have applied to Medical schools in a country could be categorised as both educational and medical; and an image that captures a beach at sunset could belong to both beach and sunset groups. Thus, all these examples naturally yield multiple labels.

Existing approaches in multi-label learning can be categorised into two main types. Problem transformation approaches decompose multi-label data into several binary problems, in order to use a binary classifier. For example, a multi-label problem with $|L|$ labels can be solved with $|L|$ binary classifiers, in which all predictions are then merged to produce final predictions. In the second type of approaches the algorithms handle multi-label data directly.

[1] Intelligent System Laboratory, University of Bristol, United Kingdom, emails: ra12404, meelis.kull, peter.flach@bristol.ac.uk
[2] King Abdulaziz University, Saudi Arabia
[3] University of Tartu, Estonia

The main challenge to note is that labels in real-world applications often have a relationship or connection, whereby the presence of one label affects or depends on another. Several studies argue that exploiting label correlations is important in the area of multi-label classification [7, 27, 28]. Although a considerable amount of work has been done in this area, these have chiefly focused on a global approach, in which label correlations are identified as a pre-processing step prior to training the model.

Decision tree algorithms are among the most widely used algorithms for classification [13, 17]. Considering the advantages of decision tree models, this paper proposes **LaCovaC**, which is a multi-label decision tree classifier. **LaCovaC** utilises the label correlation matrix at every node of the tree to find possible clusters among the labels. In addition to internal nodes that split the dataset horizontally based on selected features, **LaCovaC** introduces a second kind of node for splitting the label space vertically. At deployment, a horizontal split routes to exactly one child node according to a feature value as per normal, while a vertical split tests all outgoing edges to collect predictions about the entire labelset.

The remainder of this paper is organised as follows. Section 2 provides an overview of existing approaches to exploit correlations in multi-label classification. **LaCovaC** is presented in Section 3, and an experimental evaluation is presented and discussed in Section 4. Section 5 concludes the paper by stating the main findings and possible avenues for further work.

2 Related Work

Two well-known baseline algorithms have been considered for use as problem transformation methods: Binary Relevance (BR) [19] and Label Powerset (LP) [23]. BR applies one binary classifier to each individual label. It transforms the original dataset D into $|L|$ datasets, each of which comprises all examples of the original dataset. The examples are labelled positively if the labelset for the original example contains this label, and negatively if not. To classify a new instance, BR outputs the union of labels that have been positively predicted by the $|L|$ classifiers. Label Powerset (LP), also known as Label Combination, considers each unique set of labels that exists in a multi-label training set as a new class in a multi-class classification task. It is apparent that BR does not model label dependency, whereas LP does. However, overfitting and the exponential number of label combinations are potential difficulties affecting LP. Therefore, many different directions have been taken in the literature to address label correlations. We summarise these into several distinct approaches below.

The first approach is to transform a multi-label problem into several binary classification tasks by considering label correlations. A well-known algorithm called the Classifier Chain (CC) involves $|L|$ binary classifiers as in BR, but orders them along a chain, wherein

each classifier deals with the binary relevance problem associated with a label. Importantly, the feature space of each link in the chain is extended with the 0/1 label predictions for all previous links [16].

The same authors of CC have proposed the Ensembles of Classifier Chains (ECC), an ensemble method where the individual classifiers are trained with different orders of labels in the chain and a random subset of the training data, encouraging variability among the classifiers. These predictions are summed per label so that each label receives a number of votes. A threshold is used to select the most popular labels, which form the final predicted label set [16].

Clearly, CC and ECC are sensitive to the label order in the training process. A number of extensions to the original CC method have been proposed in [6, 7, 11, 15, 26], aiming at eliminating the key drawback in the CC, which is the lack of a principled way to decide on the label ordering.

Although CC considers label correlations by inserting the labels as new features, there is an important point to consider. It is the fact that all labels are inserted as additional features along the chain until the last label in the chain contains all previous labels as extra input features. This can be a limiting factor in high-dimensional label spaces. Furthermore, these methods force all preceding labels to be additional features for the examined label which might not be relevant or useful.

The second approach is to exploit correlations between labels by clustering them. The hierarchy of multi-label classifier (Homer) organises all labels into a tree-shaped hierarchy, with leaf nodes containing a single label [21]. Each node has training instances that are annotated with at least one of its labels. In the training phase, a multi-label classifier is trained for each node to predict a subset of labels in that node. In particular, a leaf node constructs a binary classifier to predict its single label. Given an unseen instance, Homer starts from the root node and proceeds to any successor node only if at least one of its labels was predicted by its parent node. In the end, this process reaches a subset of leaves and the final prediction is combined from predictions of these leaves. A recent work proposed in [4] combines the LP and BR methods, and is called LPBR. Its first step is to explore the dependencies between labels and then to cluster these labels into several independent subsets, according to the chi-squared statistic. Subsequently, a multi-label classifier is learnt: if the set contains only one label, BR is applied; for a group of dependent labels, LP is used. LPBR implements a greedy clustering algorithm that continues clustering as long as the loss function improves. While LPBR showed an improvement in terms of classification accuracy, it is computationally expensive.

In [12] the authors propose a method ML-LOC which first clusters the instances with respect to similarity of features and labels jointly. Subsequently, the feature space is extended with an additional feature encoding the cluster membership. Given a test instance, this additional feature is predicted using a regression model. The final predictions are then obtained from any standard multi-label classifier, trained on the extended feature space.

The final approach is to adopt decision tree algorithms in a multi-label setting. ML-C4.5 was proposed by [5] to deal with multi-label data, while the basic strategy was to define multi-label entropy separately over a set of multi-label examples. The modified entropy sums the entropies for each individual label. Another recent work was proposed in [10], and also builds a single tree for a multi-label dataset. They proposed a hybrid decision tree architecture to utilise support vector machines (SVMs) at its leaves. This approach, known as ML-SVMDT, combines two models: ML-C4.5 and BR. It builds a single decision tree, similar to ML-C4.5, whose leaves contain BR classifiers giving multi-label predictions using SVM. LaCova was proposed in [2] and is a tree based multi-label classifier that uses label covariance as a splitting criterion. The principle of LaCova is to use the label covariance matrix at each node of the tree to treat labels independently (i.e., learn a BR model from then on) or keep them together (LP) for now.

In this work we explore the value of mediating between these extreme decisions at each node in the tree. We propose **LaCovaC** which – different from previous methods – clusters labels *dynamically* during the construction of the decision tree, and hence models conditional label correlations at every node of the tree. Although other models attempt to cluster labels, they do so over the entire dataset, e.g. Homer and LPBR. In practice, conditional correlations that are used for clustering may be local, and depend on specific feature values. Hence **LaCovaC** will not separate labels automatically as happens in BR, but only in cases when labels are uncorrelated. Additionally, **LaCovaC** would not model the joint distribution at all times, as this can cause overfitting, as in LP. These decisions are taken locally at every node in the tree.

3 The Proposed Model

The key theoretical underpinning of decision trees is that they identify regions in instance space with low label variance, which is built into the splitting criterion. Naive extensions to the multi-label setting would either lead to a separate tree for each label (as in BR) or keeping all labels together (as in LP). Learning a separate tree for each label would result in as many trees as there are labels, which can be hundreds or thousands in some domains (in our experiments it can be up to 374). Furthermore, the explanatory power of the decision tree models would be reduced, which is an important factor in medical domains, among many others.

3.1 An Illustrative Example

Before providing a formal definition, a simple example is introduced here to illustrate the proposed algorithm. IMDB[4] is a multi-label dataset that contains keywords describing movies, and the classification task is to predict the movie's genre. Each movie can be assigned multiple genres from among 27 labels. For simplicity, we have selected two input features: `dark` and `love`; and three movie genres: `crime`, `horror` and `drama`.

Table 1: A small multi-label dataset with 12 instances as might be used in movie genre classification.

	labels			features	
	`crime`	`horror`	`drama`	`love`	`dark`
1	1	1	1	1	1
2	1	0	0	0	1
3	1	0	1	1	1
4	1	0	0	0	1
5	1	0	1	1	1
6	0	1	0	0	1
7	0	1	1	1	1
8	0	1	0	0	1
9	0	1	1	1	1
10	1	1	1	1	0
11	1	0	0	0	0
12	0	1	1	1	0

Inspired by the IMDB dataset we have created a toy dataset, shown in Table 1, which we use to demonstrate the advantages of **LaCovaC**

[4] http://meka.sourceforge.net/

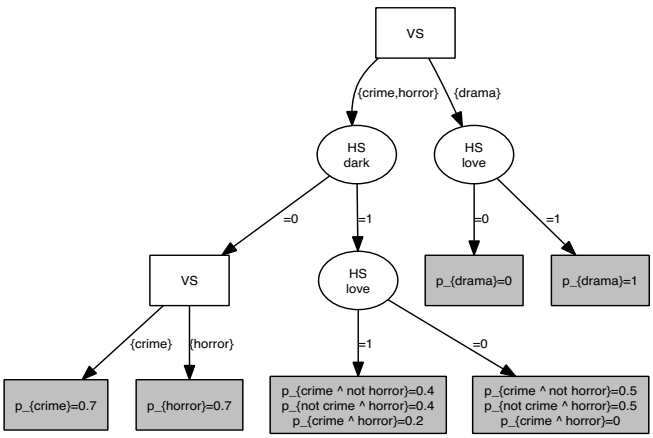


Figure 1: An example of a tree learned by **LaCovaC**. VS and HS stand for vertical and horizontal splits, respectively. The leaves show the probabilities of each label subset, except for the empty subset which can be calculated as 1 minus the sum of the given probabilities.

over binary relevance. Figure 1 shows the tree generated using **LaCovaC**. In this example, **LaCovaC** initially finds two label clusters: {crime, horror} and {drama}. Therefore, it splits the dataset vertically, and recursively creates a tree for each cluster of labels. On the left-hand side of the tree a horizontal split can be observed at the next level using the feature dark, leading to another vertical split on the left and a node that requires further horizontal splitting. Note that the second vertical split did not split further due to a minimum cardinality constraint (2 instances in this toy example). As can be seen on the left-hand side of the tree, {crime, horror} are independent when dark is false, and dependent otherwise.

When the feature love is true both labels share the same estimated marginal probability of 0.6, which can lead to predicting both of them as relevant labels, assuming 0.5 as a threshold. However, the proposed model suggests that the two subsets {crime=0, horror=1} or {crime=1, horror=0} with the highest probability among other subsets can lead to a better prediction. In other words, leaves predict the label subset that has the highest probability among others, regardless of their marginal probabilities. With regards to the third label {drama}, the proposed algorithm learns it separately as can be seen on the right-hand side of the tree.

This example demonstrates the power of modelling label correlations locally in the tree instead of globally in pre-processing, and also that the tree model represents such local correlations in a declarative way.

When applying the learned model on a test instance, one outgoing branch of a horizontal split should be followed based on the feature value as in a standard decision tree. In contrast, in a vertical split, all outgoing branches should be followed to collect predictions for all labels reaching the node. For example, in order to label a movie which has features {dark=1, love=1}, the two branches of the first vertical split should be visited. On the right-hand side of the tree, we find that the drama label does apply. On the left-hand side of the tree, we will have a horizontal split based on the dark feature leading to the second horizontal node based on the love feature. The right leaf of this second horizontal split suggests two label subsets: {crime=1, horror=0} or {crime=0, horror=1}. Suppose that the first set is chosen, then the overall predictions for this movie will be the following labels {crime=1, horror=0, drama=1}.

Figure 2 shows the result of binary relevance on this example with single-label decision trees as base model. The leftmost tree is for the crime label; the root node splits on the feature dark, when dark=0 it leads to a leaf due to a minimum cardinality constraint (2 instances). On the right-hand side of the tree, it splits further based on the feature love. Leaves show the estimated probabilities and the predicted label. The middle and right-most tree were constructed using the same method to learn labels horror and drama, respectively.

In order to predict genres for a new movie with {dark=1 and love=1} all three trees should be tested. The prediction for this movie will be {crime, horror, drama}, using 0.5 as a threshold. This is different from what the proposed model **LaCovaC** predicts.

3.2 The Main Algorithm

LaCovaC implements three key decisions at every node of the decision tree, and the process can be summarised as follows. Firstly, if labels are pure or the set of instances reaches the minimum number of data points, the algorithm stops and returns a leaf. Secondly, if the label's correlation matrix suggests the presence of cluster structure, the algorithm splits the dataset reaching that node according to the label clusters (vertical split). Finally, a set of labels located together at a node are learned, and the best features to split are determined (horizontal split).

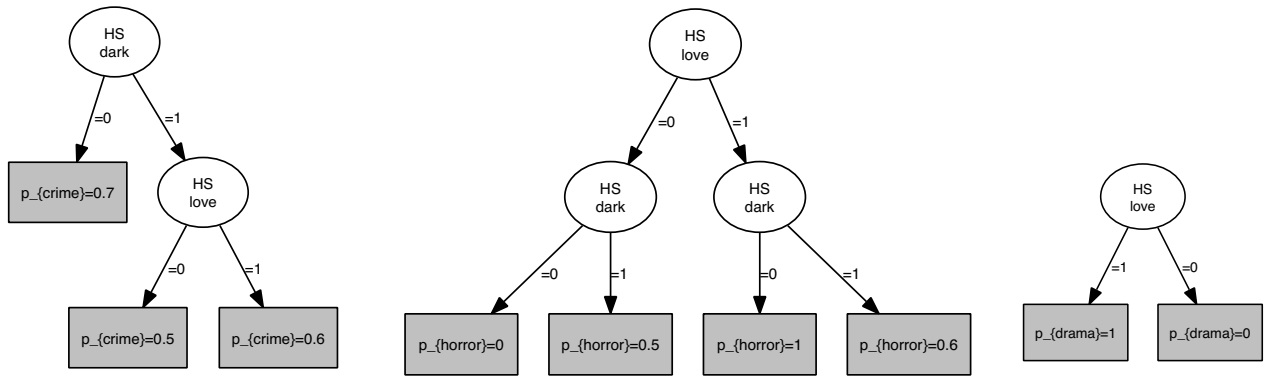
The first decision requires a threshold on the label variance. In our experiments we found that, in combination with a minimum number of instances at a leaf, a variance threshold of 0 (i.e. all labels are pure) works well. The detection of cluster structure is presented in the next section. The third decision that splits instance space horizontally also demands a splitting criterion. The most popular splitting criteria in standard decision trees learning are Gini-split (which uses the Gini index to measure impurity) and information gain (which uses Shannon entropy). However, **LaCovaC** implements a splitting criterion designed specifically for multi-label data, following a previous work [2], which can be summarised as follows. It evaluates for each child both its sum of label variances and its sum of absolute label covariances, and assigns as a splitting measure the minimum of these two (low is better). Then, it selects the feature that has the lowest splitting measure. This criterion can identify regions with either low multi-label variance or low label covariance. We improve on this by clustering the labels in a principled way into independent groups of correlated labels.

Algorithm 1 details the main algorithm implementing the approach described here.

3.3 Label Clustering

The labels are clustered based on the correlation matrix *Cor*, which is a square $|L|$ by $|L|$ matrix that contains the Pearson correlation coefficients cor_{jk} between pair of labels l_j and l_k. As labels are binary in the multi-label setting, the Pearson coefficient is reduced to the Phi coefficient, a well-known measure of association between two binary (dichotomous) variables [3, 24]. Whenever the absolute value of correlation between two labels is greater than a threshold $\lambda_{|D|}$ calculated from the number of instances (derivation given below), we decide that these labels are correlated and should be in the same cluster.

Algorithmically, this can be achieved by single linkage agglomerative hierarchical clustering as shown in Algorithm 2. The algorithm starts by creating a separate cluster for each label. It then proceeds



(a) `crime` (b) `horror` (c) `drama`

Figure 2: A separate decision tree for each label as learned by the binary relevance baseline method.

Algorithm 1 LaCovaC (D,L): Learn a tree-based multi-label classifier from training data.

Input: Dataset D; Labelset L; Minimum number m of instances to split
Output: Tree-based multi-label classifier
if Labels are pure or $|D| < m$ **then**
  Return Leaf with relative frequencies of labelsets, predicted labelset that has the highest probability
else
  $clust$=**CLUST**(D,L)
  if $|clust| > 1$ **then**
    for each set s in $clust$ **do**
      $D_s =$ Split D vertically based on the label cluster s
      /* Learn a decision tree for the label cluster s */
      $T_s =$ **LaCovaC** (D_s,s).
    end for
    Return Node splitting on $clust$ with subtrees T_s
  else
    $f, \{D_i\} =$ SelectBestFeature(D)
    for each child node D_i **do**
      $T_i =$ **LaCovaC** (D_i,L)
    end for
    Return Node splitting on f with subtrees T_i
  end if
end if

Algorithm 2 CLUST(D,L): Cluster labels based on the correlation matrix

Input: Dataset D; a set of labels $L = l_1, \cdots, l_{|L|}$;
Output: *currClust* - the clusters of labels
/* Compute the Pearson correlation coefficients between each pair of labels.*/
Cor = Correlation Matrix
/* Compute the distance between each pair of labels using the absolute correlations.*/
$Dist = 1 - |Cor|$
/* Build initial clusters with each label in a separate cluster.*/
$currClust = \{l_1\}, \{l_2\}, \cdots, \{l_{|L|}\}$
/* Create a list of label pairs sorted in ascending order of distance value.*/
$pairList =$ all pairs of labels (l_j, l_k) with $dist_{jk} < 1 - \lambda_{|D|}$, sorted by $dist_{jk}$ ascendingly
for each label pair (l_j, l_k) in pairList **do**
  if labels l_j and l_k are in different clusters **then**
    merge clusters containing l_j and l_k within *c*urrClust
  end if
end for
Return *currClust*

iteratively by taking pairs of labels in the order of increasing distance between them, where distance is defined as one minus absolute correlation, $dist_{jk} = 1 - |cor_{jk}|$. For each pair of labels, the clusters containing these labels are merged (unless they belong to the same cluster already). Such merges are performed until the distance between labels becomes greater than $1 - \lambda_{|D|}$ (that is, absolute correlation less than $\lambda_{|D|}$).

We now derive the threshold λ_n to decide whether a pair of labels are correlated or not, before merging them into one cluster. The idea is to set this threshold equal to two standard deviations in the distribution of correlation under the assumption of independent labels, hence enclosing a 95% confidence interval. The threshold depends on n, which is the number of instances reaching the current node in the tree. To derive the threshold we consider two labels with empirical frequencies p_j and p_k, respectively. The empirical Pearson correlation between these labels as Bernoulli variables can be calculated as follows:

$$\widehat{cor_{jk}} \approx \frac{C_{jk}/n - p_j p_k}{\sqrt{p_j p_k (1-p_j)(1-p_k)}}, \tag{1}$$

where C_{jk} is the the number of instances with both of those labels, and hence C_{jk}/n is the proportion of such instances. The terms $p_j(1-p_j)$ and $p_k(1-p_k)$ in the denominator are the variances of these Bernoulli variables.

Next, we study the distribution of $\widehat{cor_{jk}}$ under the assumption that labels l_j and l_k are independent. This distribution can be generated by randomly reassigning one of those labels between the instances, keeping the total frequency constant. In such case C_{jk} has a hypergeometric distribution, and we can then directly calculate its mean μ and variance σ^2:

$$\begin{aligned} C_{jk} &\sim Hypergeometric(np_j, np_k, n) \\ \mu &= np_j p_k \\ \sigma^2 &= \frac{n^2}{n-1} p_j p_k (1-p_j)(1-p_k) \end{aligned}$$

From this we can calculate the mean and variance of the correlation $\widehat{cor_{jk}}$ between labels l_j and l_k, as follows:

$$mean(\widehat{cor_{jk}}) = \frac{np_jp_k/n - p_jp_k}{\sqrt{p_jp_k(1-p_j)(1-p_k)}} = 0$$

$$var(\widehat{cor_{jk}}) = \frac{\frac{n^2}{n-1}p_jp_k(1-p_j)(1-p_k)/n^2}{p_jp_k(1-p_j)(1-p_k)} = \frac{1}{n-1}$$

We define λ_n as the value enclosing a 95% confidence interval assuming a normal distribution for $\widehat{cor_{jk}}$ which can be calculated in the usual way:

$$\lambda_n = 1.96 \cdot \sqrt{var(\widehat{cor_{jk}})} = \frac{1.96}{\sqrt{n-1}} \quad (2)$$

where n is the number of instances reaching a particular decision node in the tree. Note that this threshold is always less than 1 as $n > 5$ in our experiments.

4 Experimental Evaluation

In total, 13 common benchmarks were retrieved for use from the Meka[5] and Mulan [22] repositories. The key statistics of these datasets are shown in Table 2.

Table 2: The statistics for the datasets used in the experiments. $|L|$ is the number of labels, $|D|$ is the number of instances, *att* is the number of attributes (features), *card* is the average number of labels per instance, and *dens* is the label density.

	$\|L\|$	$\|D\|$	*att*	*card*	*dens*
Corel5k	374	5000	499	3.522	0.94%
Cal500	174	502	68	26.044	14.96%
Bibtex	159	7395	1836	2.402	1.51%
Language log	75	1460	1004	1.179	1.57%
Enron	53	1702	1001	3.378	6.37%
Medical	45	978	1449	1.245	2.76%
Genebase	27	662	1186	1.252	4.63 %
Slashdot	22	3782	1079	1.180	5.36%
Birds	19	645	260	1.014	5.30%
Yeast	14	2417	103	4.237	30.26%
Flags	7	194	19	3.392	48.45%
Emotions	6	593	66	1.869	31.14%
Scene	6	2407	294	1.074	17.89%

4.1 Experimental Setup

BR, LP, and CC algorithms were run in Meka, whereas Mulan was used for the Homer and LPBR algorithms. When employing all these methods, the trees were produced with Weka's J48 algorithm [6]. The Homer algorithm was run using the best setting, as reported by [21]. LPBR requires parameter configurations, such as non-improving counter to prevent clustering. The default parameters settings in Mulan were 5 for the non-improving counter and a 5-fold cross validation for testing the clustering performance. The target loss function to evaluate the clustering was set to exact-match according to Mulan. Exact-match is a strict measure that examines whether the predicted and actual label sets are equal or not.

LaCovaC was implemented in Java on the Meka platform. The LaCova and ML-C4.5 implementations are provided in [2]. The minimal number of instances in the leaves in ML-C4.5, LaCova, and **LaCovaC** was set to 5.

10-fold cross-validation was performed throughout except the two largest datasets in terms of both number of labels and number of instances: Corel5k and Bibtex. The exception is because cross validation was intensive for LPBR as we were not able to run this algorithm using the available resources [7]. Therefore, a train/test split was used instead, which was provided by Mulan.

For all methods we used the Pcut method to convert the label probabilities into relevant labels [1, 25]. This method uses a single threshold for all labels, chosen such that the average label density in test data is the same as in the training data.

4.2 Evaluation Metrics

All algorithms were evaluated under the Meka framework. We used the most common multi-label metrics, namely multi-label accuracy (Jaccard index), exact-match (subset accuracy), Hamming loss and three versions of the F_1 score: micro-averaged over all instance-label combinations, macro-averaged per label, and macro-averaged per instance. We also used a multi-label version of log-loss [16] to evaluate the predicted probabilities. Log-loss evaluates a classifier's confidence by punishing errors with higher probability more severely. We used a dataset-dependent maximum of $\log(\frac{1}{|D|})$ to limit the magnitudes of penalty as suggested in [16]. The description and formal definition of these evaluation measures for multi-label data are given below.

Multi-label accuracy calculates the ratio of the union and intersection of the predicted and actual labelsets, averaged over all examples [28]:

$$mla = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap \hat{Y}_i|}{|Y_i \cup \hat{Y}_i|} \quad (3)$$

where Y_i and $\hat{Y}_i$ are the set of actual labels and predicted labels for an instance, respectively.

Exact-match examines if the predicted and actual label subsets are equal or not [9]:

$$em = \frac{1}{|D|}\sum_{i=1}^{|D|} I(Y_i = \hat{Y}_i) \quad (4)$$

where $I(true) = 1$ and $I(false) = 0$.

Hamming loss takes the proportion of misclassified labels (labels predicted incorrectly and correct labels that are not predicted) averaged over all examples [18]:

$$hl = \frac{1}{|L| * |D|}\sum_{i=1}^{|D|}|Y_i \triangle \hat{Y}_i| \quad (5)$$

where $\triangle$ is the symmetric difference.

[5] http://meka.sourceforge.net/
[6] http://sourceforge.net/projects/weka
[7] 2.7 GHz Intel Core i5 processor and 8GB of memory

Log-loss evaluates a classifier's confidence by punishing errors with higher probability more severely [16]:

$$ls = \frac{1}{|D|}\sum_{i=1}^{|D|}\sum_{j=1}^{|L|}\log(pr_{ij})\cdot y_{ij} + \log(1-pr_{ij})\cdot(1-y_{ij}) \quad (6)$$

where y_{ij} indicates whether the j^{th} label is relevant for the i^{th} instance (value 1) or irrelevant (value 0). pr_{ij} is the probability estimate for the i^{th} instance and the j^{th} label.

Micro F_1 corresponds to the harmonic mean of precision and recall [14]. It put all predictions on all labels in one vector as in binary classification and then calculates the F_1:

$$\text{micro } F_1 = \frac{\sum_{i=1}^{|D|}\sum_{j=1}^{|L|} 2\cdot y_{ij}\cdot\hat{y}_{ij}}{\sum_{i=1}^{|D|}\sum_{j=1}^{|L|}(y_{ij}+\hat{y}_{ij})} \quad (7)$$

Macro f_l is the averaged F_1 score over labels [14]:

$$\text{macro } f_l = \frac{1}{|L|}\sum_{j=1}^{|L|} F_1^{(j)} \quad (8)$$

where $F_1^{(j)}$ is the F_1 score for the j^{th} label vector.

Macro f_e is the averaged F_1 score over examples [14]:

$$\text{macro } f_e = \frac{1}{|D|}\sum_{i=1}^{|D|} F_1^{i} \quad (9)$$

where F_1^i is the F_1 score for the i^{th} instance row vector.

4.3 Results and Discussion

We conducted the Friedman test based on the average ranks for all datasets [8]. This test ranks the algorithms for each dataset separately, thus the best algorithm gets the rank of 1, the second best the rank of 2, etc. Then, it calculates the test statistic on the ranks averaged over all datasets in order to verify whether the differences between algorithms are statistically significant.

Table 3 shows the average ranks for each algorithm over 13 datasets and seven evaluation measures. We also show the mean of these average ranks aggregated over all evaluation measures, which shows that overall **LaCovaC** scores highest, followed by CC and BR. The Friedman test gave a significant difference at 5% confidence for all metrics except multi-label accuracy; therefore, we carried out post-hoc Nemenyi tests as shown in Figure 3.

It can be seen that **LaCovaC** outperforms Homer, ML-C4.5, and LPBR in the average ranks with respect to all the evaluation measures used in this paper, and in several cases statistically significantly. **LaCovaC** is not significantly worse than the best performing algorithm (LP) in terms of exact-match. In fact, according to Table 4 **LaCovaC** loses to LP in the 4 datasets with the smallest numbers of labels (up to 14), but wins in 6 out of the 9 datasets with more labels. This fact confirms the assumption that LP can overfit the training data in case of large number of labels and label combinations because it can only model labelsets observed in the training. Notably, for exact-match the proposed algorithm **LaCovaC** is the overall best method or tied with the best in the three datasets with the biggest numbers of labels (Corel5k, Cal500 and Bibtex with 374, 174 and 159 labels, respectively). The proposed algorithm has better Hamming loss than LP in 9 out of 13 datasets.

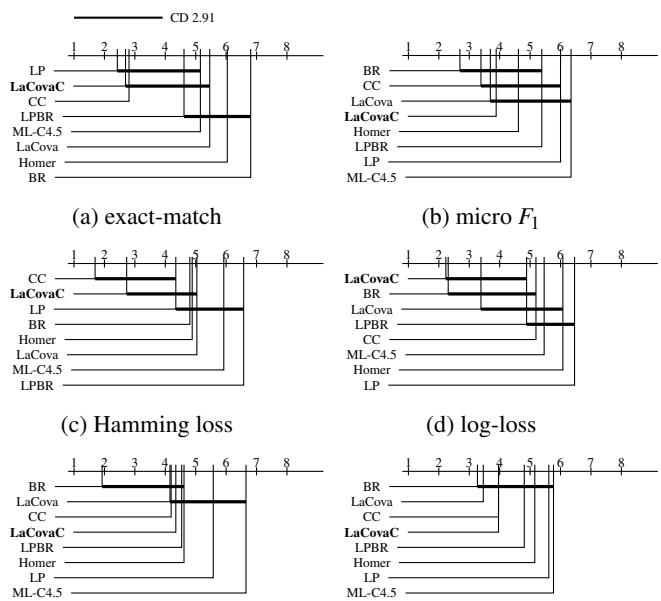
Additionally, **LaCovaC** outperforms BR in terms of exact-match, log-loss and Hamming loss. BR has the best average rank considering F_1 score per instance average f_e, F_1 per label average f_l and micro F_1 as it decomposes the multi-label problem into several binary classifiers, which can be better for F_1 score.

We can further observe that **LaCovaC** has the best average rank in terms of log-loss that evaluates the classifiers' scores regardless of the thresholding technique. It is significantly better than CC, ML-C4.5, Homer and LP.

Finally, CC gets top average rank for Hamming loss and **LaCovaC** is the second best without significant loss against CC. Detailed results of the experiments are given in Tables 4, 5 and 6.

Table 3: Average ranks obtained by the Friedman test over 13 datasets. The last column shows the mean of the average ranks obtained by each algorithm over all the evaluation measures, and the algorithms are sorted on decreasing mean. *mla*, *em*, *hl* and *ls* denote multi-label accuracy, exact-match, Hamming loss and log-loss, respectively. micro F_1 is the micro averaged $F1$. macro f_l and macro f_e are F_1 scores averaged in terms of labels and examples, respectively.

	mla	*em*	*hl*	*ls*	micro F_1	macro f_l	macro f_e	mean
LaCovaC	3.57	2.69	2.73	**2.23**	3.88	4.34	3.96	3.34
CC	**3.46**	2.8	**1.69**	5.19	3.38	4.19	3.69	3.49
BR	4.57	6.8	4.8	2.3	**2.69**	**1.92**	**3.26**	3.76
LaCova	3.65	5.46	5.03	3.38	3.69	4.15	3.46	4.12
LP	4.92	**2.42**	4.34	6.46	6	5.57	5.61	5.05
LPBR	4.84	4.61	6.57	4.88	5.38	4.53	4.8	5.09
Homer	5.3	6.03	4.88	6.07	4.61	4.61	5.15	5.24
ML-C4.5	5.65	5.15	5.92	5.46	6.34	6.65	5.76	5.85



(a) exact-match (b) micro F_1

(c) Hamming loss (d) log-loss

(e) macro F_1 with respect to labels (f) macro F_1 with respect to examples

Figure 3: Critical Difference diagrams using pairwise comparisons for experiments where the Friedman test yields significance at 0.05.

Table 4: Results on 13 datasets with regards to multi-label accuracy and exact-match, the higher the value the better.

	BR	LP	CC	Homer	LPBR	ML-C4.5	LaCova	**LaCovaC**
	multi-label accuracy							
Corel5k	0.089(2)	0.080(7)	0.083(6)	**0.092(1)**	0.023(8)	0.087(4)	0.084(5)	0.088(3)
Cal500	0.201(6)	0.204(5)	0.208(4)	0.197(7)	0.067(8)	**0.278(1)**	0.254(2)	0.209(3)
Bibtex	0.259(5)	0.244(6)	0.276(3)	0.208(7)	**0.296(1)**	0.176(8)	0.267(4)	0.280(2)
Language log	0.245(2)	0.234(4)	**0.247(1)**	0.233(5)	0.184(8)	0.224(7)	0.235(3)	0.232(6)
Enron	0.296(3)	0.277(5)	0.315(2)	0.281(4)	**0.320(1)**	0.207(8)	0.256(6)	0.246(7)
Medical	0.729(4)	0.724(6)	0.728(5)	0.700(7)	0.745(2)	0.363(8)	0.732(3)	**0.762(1)**
Genebase	0.901(8)	0.980(3)	**0.986(1)**	0.952(5)	0.955(4)	0.946(7)	0.950(6)	0.981(2)
Slashdot	0.425(4)	0.420(5)	0.429(3)	0.379(6)	0.376(7)	0.372(8)	0.439(2)	**0.449(1)**
Birds	0.491(3)	0.453(7)	0.486(4)	0.493(2)	0.411(8)	0.466(6)	0.481(5)	**0.499(1)**
Yeast	0.391(6.5)	0.396(4.5)	0.397(3)	0.381(8)	0.411(2)	0.396(4.5)	**0.416(1)**	0.391(6.5)
Flags	0.534(6)	0.531(7)	0.540(5)	0.544(4)	0.509(8)	**0.572(1)**	0.556(3)	0.561(2)
Emotions	0.402(4)	0.416(2)	0.388(7)	0.390(6)	**0.417(1)**	0.406(3)	0.400(5)	0.384(8)
Scene	0.403(6)	0.432(2.5)	**0.464(1)**	0.380(7)	0.421(5)	0.366(8)	0.432(2.5)	0.429(4)
Average ranks	4.57	4.92	**3.46**	5.30	4.84	5.65	3.65	3.57
	exact-match							
Corel5k	0.000(8)	0.008(2)	0.006(3)	0.004(4.5)	0.002(6.5)	0.002(6.5)	0.004(4.5)	**0.018(1)**
Cal500	**0.000(4.5)**	**0.000(4.5)**	**0.000(4.5)**	**0.000(4.5)**	**0.000(4.5)**	**0.000(4.5)**	**0.000(4.5)**	**0.000(4.5)**
Bibtex	0.079(8)	0.137(2)	0.122(4)	0.084(7)	0.123(3)	0.092(6)	0.108(5)	**0.149(1)**
Language log	0.184(5)	**0.205(1)**	0.203(2)	0.183(6)	0.153(8)	0.195(4)	0.177(7)	0.197(3)
Enron	0.022(7)	**0.079(1)**	0.076(2)	0.031(5.5)	0.045(3)	0.020(8)	0.031(5.5)	0.032(4)
Medical	0.607(6)	0.642(3)	0.641(4)	0.598(7)	0.655(2)	0.293(8)	0.625(5)	**0.673(1)**
Genebase	0.809(8)	0.962(3)	**0.971(1)**	0.921(6)	0.938(4)	0.923(5)	0.914(7)	0.967(2)
Slashdot	0.296(7)	0.368(2)	0.356(3)	0.262(8)	0.306(6)	0.308(5)	0.353(4)	**0.392(1)**
Birds	0.398(4)	0.376(7)	**0.410(1)**	0.404(3)	0.339(8)	0.379(6)	0.395(5)	0.407(2)
Yeast	0.035(8)	**0.134(1)**	0.118(4)	0.050(7)	0.122(3)	0.109(5)	0.107(6)	0.124(2)
Flags	0.119(8)	0.202(2)	0.182(3)	0.155(5)	0.145(6)	**0.207(1)**	0.135(7)	0.171(4)
Emotions	0.125(8)	**0.202(1)**	0.157(4)	0.142(7)	0.165(3)	0.179(2)	0.143(5.5)	0.143(5.5)
Scene	0.284(7)	0.393(2)	**0.416(1)**	0.277(8)	0.375(3)	0.307(6)	0.354(5)	0.366(4)
Average ranks	6.80	**2.42**	2.80	6.03	4.61	5.15	5.46	2.69

Table 5: Results on 13 datasets with regards to Hamming loss and log-loss, the lower the value the better.

	BR	LP	CC	Homer	LPBR	ML-C4.5	LaCova	**LaCovaC**
	Hamming loss							
Corel5k	0.017(5.5)	0.016(3)	**0.011(1)**	0.016(3)	0.023(7)	0.203(8)	0.016(3)	0.017(5.5)
Cal500	0.223(7)	0.201(5)	0.189(2.5)	0.204(6)	0.540(8)	**0.188(1)**	0.189(2.5)	0.198(4)
Bibtex	0.022(5.5)	0.020(4)	**0.017(1)**	0.022(5.5)	0.018(2.5)	0.026(8)	0.024(7)	0.018(2.5)
Language log	0.031(6)	0.026(3)	**0.023(1)**	0.027(4)	0.069(8)	0.029(5)	0.033(7)	0.025(2)
Enron	0.082(6)	0.078(4.5)	**0.062(1)**	0.078(4.5)	0.077(2.5)	0.202(8)	0.085(7)	0.077(2.5)
Medical	0.013(3.5)	0.014(5.5)	0.011(2)	0.016(7)	0.013(3.5)	0.163(8)	0.014(5.5)	**0.010(1)**
Genebase	0.008(4.5)	0.002(2.5)	**0.001(1)**	0.008(4.5)	0.019(7.5)	0.019(7.5)	0.012(6)	0.002(2.5)
Slashdot	0.055(3)	0.058(4.5)	0.045(2)	0.075(7)	0.086(8)	0.066(6)	0.058(4.5)	**0.028(1)**
Birds	0.072(3)	0.076(5.5)	**0.063(1.5)**	0.075(4)	0.113(8)	0.077(7)	0.076(5.5)	**0.063(1.5)**
Yeast	0.296(7)	0.288(4)	0.281(2)	0.290(6)	0.363(8)	0.289(5)	**0.276(1)**	0.285(3)
Flags	0.307(6)	0.309(7)	0.302(5)	0.301(4)	0.332(8)	**0.273(1)**	0.293(3)	0.280(2)
Emotions	0.296(2)	0.304(3)	**0.286(1)**	0.305(4.5)	0.309(7.5)	0.305(4.5)	0.309(7.5)	0.306(6)
Scene	0.193(3.5)	0.200(5)	**0.185(1)**	0.193(3.5)	0.238(7)	0.294(8)	0.201(6)	0.188(2)
Average ranks	4.80	4.34	**1.69**	4.88	6.57	5.92	5.03	2.73
	log-loss							
Corel5k	**0.048(1)**	0.101(5)	0.070(4)	0.206(6)	0.275(8)	0.216(7)	0.064(3)	0.061(2)
Cal500	0.481(4)	0.786(7)	0.741(6)	0.586(5)	0.853(8)	**0.381(1)**	0.407(3)	0.394(2)
Bibtex	**0.068(1)**	0.158(6)	0.132(5)	0.194(7)	0.075(3)	0.220(8)	0.078(4)	0.070(2)
Language log	**0.080(1)**	0.130(6)	0.113(4)	0.165(7)	0.122(5)	0.349(8)	0.085(2)	0.093(3)
Enron	**0.197(1)**	0.402(7)	0.353(5)	0.367(6)	0.230(3)	0.437(8)	0.261(4)	0.218(2)
Medical	**0.035(1)**	0.064(5)	0.052(3.5)	0.084(7)	0.052(3.5)	0.364(8)	0.066(6)	0.037(2)
Genebase	0.008(3)	0.007(2)	**0.005(1)**	0.022(6)	0.012(4)	0.064(8)	0.032(7)	0.017(5)
Slashdot	**0.163(1)**	0.346(7)	0.269(5)	0.333(6)	0.241(4)	0.368(8)	0.204(3)	0.171(2)
Birds	0.181(2)	0.262(8)	0.217(5)	0.238(6)	0.215(4)	0.239(7)	0.190(3)	**0.180(1)**
Yeast	0.671(4)	1.583(8)	1.540(7)	0.900(6)	0.799(5)	0.585(2)	**0.566(1)**	0.603(3)
Flags	0.619(4)	0.918(8)	0.897(7)	0.716(6)	0.713(5)	**0.546(1)**	0.616(3)	0.559(2)
Emotions	0.640(4)	1.243(7)	1.245(8)	0.832(6)	0.815(5)	**0.603(1)**	0.630(3)	0.619(2)
Scene	0.492(3)	1.095(8)	1.014(7)	0.726(5)	0.733(6)	0.530(4)	0.482(2)	**0.475(1)**
Average ranks	2.30	6.46	5.19	6.07	4.88	5.46	3.38	**2.23**

Table 6: Results on 13 datasets with regards to micro F_1, macro f_l and macro f_e, the higher the value the better.

	BR	LP	CC	Homer	LPBR	ML-C4.5	LaCova	**LaCovaC**
	micro F_1							
Corel5k	0.149(2)	0.114(6)	0.148(3)	**0.154(1)**	0.038(7)	0.031(8)	0.141(4)	0.126(5)
Cal500	0.334(5)	0.332(6)	0.338(4)	0.328(7)	0.182(8)	**0.420(1)**	0.397(2)	0.341(3)
Bibtex	0.345(4)	0.285(6)	0.361(2)	0.272(7)	**0.391(1)**	0.195(8)	0.311(5)	0.351(3)
Language log	0.175(2)	0.122(7)	**0.177(1)**	0.158(3)	0.123(6)	0.106(8)	0.152(4)	0.142(5)
Enron	0.428(2)	0.364(7)	0.426(3)	0.399(4)	**0.442(1)**	0.313(8)	0.373(5)	0.366(6)
Medical	0.778(3)	0.744(6)	0.786(2)	0.726(7)	0.774(4)	0.355(8)	0.761(5)	**0.806(1)**
Genebase	0.921(4)	0.982(2)	**0.988(1)**	0.920(5)	0.853(7)	0.841(8)	0.893(6)	0.981(3)
Slashdot	0.506(2)	0.429(6)	**0.512(1)**	0.456(5)	0.381(8)	0.383(7)	0.468(4)	0.496(3)
Birds	**0.366(1)**	0.295(7)	0.312(6)	0.334(4)	0.281(8)	0.330(5)	0.336(3)	0.346(2)
Yeast	0.541(2)	0.522(8)	0.528(5)	0.531(3)	0.530(4)	0.525(6)	**0.549(1)**	0.523(7)
Flags	0.690(5)	0.677(7)	0.680(6)	0.696(4)	0.665(8)	**0.718(1)**	0.704(3)	0.706(2)
Emotions	0.539(2)	0.521(4)	0.499(8)	0.520(5)	**0.540(1)**	0.511(6.5)	0.526(3)	0.511(6.5)
Scene	**0.493(1)**	0.445(6)	0.487(2)	0.466(5)	0.429(7)	0.380(8)	0.473(3)	0.472(4)
Average ranks	**2.69**	6.00	3.38	4.61	5.38	6.34	3.69	3.88
	macro f_l							
Corel5k	**0.040(1)**	0.030(2)	0.027(4)	0.024(6)	0.012(8)	0.021(7)	0.029(3)	0.026(5)
Cal500	0.161(2)	0.147(6.5)	0.146(8)	0.156(3.5)	**0.168(1)**	0.147(6.5)	0.155(5)	0.156(3.5)
Bibtex	0.275(2)	0.193(6)	0.258(3)	0.147(7)	**0.280(1)**	0.118(8)	0.240(5)	0.248(4)
Language log	**0.068(1)**	0.039(6)	0.057(2)	0.050(4.5)	0.050(4.5)	0.038(7.5)	0.054(3)	0.038(7.5)
Enron	**0.132(1)**	0.090(7.5)	0.122(3)	0.123(2)	0.111(4)	0.098(5)	0.097(6)	0.090(7.5)
Medical	**0.342(1)**	0.311(6)	0.329(3)	0.304(7)	0.323(4)	0.172(8)	0.314(5)	0.334(2)
Genebase	0.518(4)	**0.556(1.5)**	**0.556(1.5)**	0.510(5)	0.504(6)	0.474(8)	0.492(7)	0.544(3)
Slashdot	**0.332(1)**	0.267(6)	0.315(2)	0.280(5)	0.235(8)	0.239(7)	0.294(4)	0.310(3)
Birds	**0.208(1)**	0.140(8)	0.143(7)	0.182(4)	0.164(5.5)	0.164(5.5)	0.190(2)	0.185(3)
Yeast	0.394(2)	0.372(7)	0.381(5)	0.389(3)	**0.406(1)**	0.359(8)	0.385(4)	0.379(6)
Flags	0.600(6)	0.601(5)	0.581(7)	0.616(3)	0.567(8)	**0.622(1)**	0.610(4)	0.620(2)
Emotions	**0.530(1)**	0.503(4)	0.488(8)	0.502(5)	0.527(2)	0.494(7)	0.511(3)	0.495(6)
Scene	0.244(2)	0.221(7)	**0.247(1)**	0.233(5)	0.223(6)	0.213(8)	0.236(3)	0.235(4)
Average ranks	**1.92**	5.57	4.19	4.61	4.53	6.65	4.15	4.34
	macro f_e							
Corel5k	0.140(2)	0.115(7)	0.121(6)	**0.143(1)**	0.032(8)	0.131(3)	0.127(4)	0.125(5)
Cal500	0.330(5)	0.327(6)	0.334(4)	0.323(7)	0.119(8)	**0.424(1)**	0.393(2)	0.335(3)
Bibtex	0.339(4)	0.292(6)	0.341(2)	0.266(7)	**0.369(1)**	0.216(8)	0.338(5)	0.340(3)
Language log	**0.133(1)**	0.104(6)	0.124(2)	0.115(4)	0.096(7)	0.094(8)	0.121(3)	0.106(5)
Enron	0.416(2.5)	0.368(5)	0.416(2.5)	0.392(4)	**0.434(1)**	0.297(8)	0.360(6)	0.342(7)
Medical	0.770(3)	0.752(6)	0.757(5)	0.735(7)	0.776(2)	0.389(8)	0.769(4)	**0.793(1)**
Genebase	0.931(8)	0.985(3)	**0.990(1)**	0.962(4)	0.961(5)	0.952(7)	0.960(6)	0.986(2)
Slashdot	**0.473(1)**	0.438(5)	0.454(4)	0.422(6)	0.408(7)	0.394(8)	0.471(2)	0.469(3)
Birds	**0.182(1)**	0.142(6)	0.138(7.5)	0.150(4)	0.138(7.5)	0.146(5)	0.172(2)	0.160(3)
Yeast	0.520(3)	0.495(8)	0.502(4)	0.500(6)	**0.526(1)**	0.501(5)	0.524(2)	0.498(7)
Flags	0.647(6)	0.636(7)	0.650(5)	0.655(4)	0.618(8)	**0.678(1)**	0.673(3)	0.674(2)
Emotions	0.500(2)	0.494(3)	0.469(7.5)	0.483(6)	**0.505(1)**	0.487(5)	0.488(4)	0.469(7.5)
Scene	0.446(4)	0.445(5)	**0.481(1)**	0.416(7)	0.441(6)	0.394(8)	0.460(2)	0.451(3)
Average ranks	**3.26**	5.61	3.96	5.15	4.80	5.76	3.46	3.96

5 Concluding Remarks

We presented a novel decision tree algorithm for multi-label classification called **LaCovaC**. The key idea of this algorithm is to compute the label correlation matrix at each node of the tree in order to identify label correlations and then cluster them locally. Its main innovation is to introduce vertical splits that separate locally independent labels, in addition to the traditional feature-based horizontal splits that divide the instance space as in the standard decision tree algorithms.

To evaluate **LaCovaC** we compared it to state-of-the-art approaches. We used seven common evaluation metrics and 13 datasets. **LaCovaC** has the best average rank for log-loss and the second best for multi-label accuracy, exact-match and Hamming loss (without significant loss). For exact-match, **LaCovaC** shows a comparable performance to the best algorithm LP, which is a strong baseline for exact-match and hard to beat. In addition, it outperforms LP on 6 of the 9 largest datasets in terms of number of labels. This suggests that combining LP and BR as in **LaCovaC** leads to improvement over those metrics and reduces the overfitting risk.

Overall, the main strength of **LaCovaC** is its strong performance across all these metrics, as can be seen in the right-most column of Table 3. Furthermore, on each of the metrics individually **LaCovaC** is not significantly worse than the top contender. It is widely recognised in the literature that different multi-label evaluation metrics measure different things; hence there is value in demonstrating that an algorithm performs well across the board.

Several directions can be taken for further work. We plan to investigate the parameter configuration for **LaCovaC**, for example, a stopping criterion for the clustering algorithm. Moreover, it would be interesting to investigate alternative clustering approaches, such as spectral clustering. Furthermore, to counteract the large variance associated with decision tree learning – which carries over to the proposed model – we could use bootstrap aggregates as in random forests. Finally, using the significance threshold on correlation balances the sample size and the strength of the correlation, such that both low correlation on a large sample size and high correlation on a small sample are detected as significant. However, finding an even better balance between effect size and sample size is an interesting future research direction.

Acknowledgments

Reem Al-Otaibi's PhD study is sponsored by King Abdulaziz University, Saudi Arabia. This work was partly supported by the REFRAME project granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences and Technologies ERA-Net (CHISTERA), and funded by the Engineering and Physical Sciences Research Council in the UK under grant EP/K018728/1. We acknowledge the comments of the anonymous reviewers which helped us improve the paper.

REFERENCES

[1] Reem Al-Otaibi, Peter Flach, and Meelis Kull, 'Multi-label classification: A comparative study on threshold selection methods', in *First International Workshop on Learning over Multiple Contexts (LMCE) at ECML-PKDD 2014*, Nancy, France, (September 2014).

[2] Reem Al-Otaibi, Meelis Kull, and Peter Flach, 'Lacova: A tree-based multi-label classifier using label covariance as splitting criterion', in *Proceedings of the IEEE 13th International Conference on Machine*

Learning and Application (ICMLA-2014), pp. 74–79, Detroit, USA, (December 2014). IEEE.

[3] Alan H. Cheetham and Joseph E. Hazel, 'Binary (Presence-Absence) Similarity Coefficients', *Journal of Paleontology*, **43**(5), 1130–1136, (1969).

[4] Lena Chekina, Dan Gutfreund, Aryeh Kontorovich, Lior Rokach, and Bracha Shapira, 'Exploiting label dependencies for improved sample complexity', *Machine Learning*, **91**(1), 1–42, (April 2013).

[5] Amanda Clare, A Clare, and Ross D. King, 'Knowledge discovery in multi-label phenotype data', in *Lecture Notes in Computer Science*, pp. 42–53. Springer, (2001).

[6] Pablo Nascimento da Silva, Eduardo Corrêa Gonçalves, Alexandre Plastino, and Alex Freitas, 'Distinct chains for different instances: An effective strategy for multi-label classifier chains', in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, volume 8725, 453–468, Springer Berlin Heidelberg, (2014).

[7] Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier, 'Bayes optimal multilabel classification via probabilistic classifier chains.', in *Proceedings of the International Conference on Machine Learning (ICML10)*, eds., Johannes Fürnkranz and Thorsten Joachims, pp. 279–286. Omnipress, (2010).

[8] Janez Demšar, 'Statistical comparisons of classifiers over multiple data sets', *Journal of Machine Learning Research*, **7**, 1–30, (December 2006).

[9] Nadia Ghamrawi and Andrew McCallum, 'Collective multi-label classification', in *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pp. 195–200, New York, NY, USA, (2005). ACM.

[10] Dejan Gjorgjevikj, Gjorgji Madjarov, and Sašo Dzeroski, 'Hybrid decision tree architecture utilizing local svms for efficient multi-label learning', *IJPRAI*, **27**(7), (2013).

[11] Eduardo Corrêa Gonçalves, Alexandre Plastino, and Alex Freitas, 'Simpler is better: A novel genetic algorithm to induce compact multi-label chain classifiers', in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pp. 559–566, New York, NY, USA, (2015). ACM.

[12] Sheng-Jun Huang and Zhi-Hua Zhou, 'Multi-label learning by exploiting label correlations locally', in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, (2012).

[13] S.B. Kotsiantis, 'Decision trees: a recent overview', *Artificial Intelligence Review*, **39**(4), 261–283, (2013).

[14] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Dzeroski, 'An extensive experimental comparison of methods for multi-label learning', *Pattern Recognition*, **45**(9), 3084–3104, (September 2012).

[15] Jesse Read, Luca Martino, and David Luengo, 'Efficient monte carlo methods for multi-dimensional learning with classifier chains', *Pattern Recognition*, **47**(3), 1535–1546, (2014).

[16] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank, 'Classifier chains for multi-label classification', in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, pp. 254–269, Berlin, Heidelberg, (2009). Springer-Verlag.

[17] Lior Rokach, 'Decision forest: Twenty years of research', *Information Fusion*, **27**, 111–125, (2016).

[18] Robert E. Schapire and Yoram Singer, 'Improved boosting algorithms using confidence-rated predictions', in *Machine Learning*, pp. 297–336, (1999).

[19] Grigorios Tsoumakas and Ioannis Katakis, 'Multi-label classification: An overview', *International Journal of Data Warehousing and Mining*, **2007**, 1–13, (2007).

[20] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas, 'Mining multi-label data', in *Data Mining and Knowledge Discovery Handbook*, pp. 667–685, (2010).

[21] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas, 'Effective and Efficient Multilabel Classification in Domains with Large Number of Labels', in *ECML/PKDD 2008 Workshop on Mining Multidimensional Data*, (2008).

[22] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas, 'Mulan: A java library for multi-label learning', *Journal of Machine Learning Research*, **12**, 2411–2414, (2011).

[23] Grigorios Tsoumakas and Ioannis Vlahavas, 'Random k-labelsets: An ensemble method for multilabel classification', in *Proceedings of the 18th European Conference on Machine Learning*, ECML07, pp. 406–417, Berlin, Heidelberg, (2007). Springer-Verlag.

[24] Matthijs J. Warrens, 'On association coefficients for 2x2 tables and properties that do not depend on the marginal distributions', *Psychometrika*, **73**(4), 777–789, (2008).

[25] Yiming Yang, 'A study of thresholding strategies for text categorization', in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pp. 137–145, New York, NY, USA, (2001). ACM.

[26] Julio H. Zaragoza, L. Enrique Sucar, Eduardo F. Morales, Concha Bielza, and Pedro Larrañaga, 'Bayesian chain classifiers for multidimensional classification', in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pp. 2192–2197. AAAI Press, (2011).

[27] Min-Ling Zhang and Kun Zhang, 'Multi-label learning by exploiting label dependency', in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pp. 999–1008, New York, NY, USA, (2010). ACM.

[28] Min-Ling Zhang and Zhi-Hua Zhou, 'A review on multi-label learning algorithms', *Knowledge and Data Engineering, IEEE Transactions on.*, **26**(8), 1819–1837, (2014).